```
RRRRRRRRRRR      MMM          MMM     SSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSSSSS
RRR         RRR  MMMMMM    MMMMMM   SSS
RRR         RRR  MMMMMM    MMMMMM   SSS
RRR         RRR  MMMMMM    MMMMMM   SSS
RRR         RRR  MMM  MMM  MMM  MMM  SSS
RRR         RRR  MMM  MMM  MMM  MMM  SSS
RRR         RRR  MMM  MMM  MMM  MMM  SSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSS
RRR   RRR        MMM          MMM             SSS
RRR   RRR        MMM          MMM             SSS
RRR   RRR        MMM          MMM             SSS
RRR      RRR     MMM          MMM             SSS
RRR      RRR     MMM          MMM             SSS
RRR      RRR     MMM          MMM             SSS
RRR         RRR  MMM          MMM  SSSSSSSSSSS
RRR         RRR  MMM          MMM  SSSSSSSSSSS
RRR         RRR  MMM          MMM  SSSSSSSSSSS
```

```
RRRRRRRR    MM      MM    333333          JJ    000000    UU      UU  RRRRRRRR    NN      NN  LL
RRRRRRRR    MM      MM    333333          JJ    000000    UU      UU  RRRRRRRR    NN      NN  LL
RR      RR  MMMM  MMMM  33      33        JJ  00      00  UU      UU  RR      RR  NN      NN  LL
RR      RR  MMMM  MMMM  33      33        JJ  00      00  UU      UU  RR      RR  NN      NN  LL
RR      RR  MM  MM  MM          33        JJ  00      00  UU      UU  RR      RR  NNNN    NN  LL
RR      RR  MM  MM  MM          33        JJ  00      00  UU      UU  RR      RR  NNNN    NN  LL
RRRRRRRR    MM      MM        33          JJ  00      00  UU      UU  RRRRRRRR    NN  NN  NN  LL
RRRRRRRR    MM      MM        33          JJ  00      00  UU      UU  RRRRRRRR    NN  NN  NN  LL
RR  RR      MM      MM          33  JJ      JJ  00      00  UU      UU  RR  RR      NN    NNNN  LL
RR  RR      MM      MM          33  JJ      JJ  00      00  UU      UU  RR  RR      NN    NNNN  LL
RR    RR    MM      MM  33      33  JJ      JJ  00      00  UU      UU  RR    RR    NN      NN  LL                ....
RR    RR    MM      MM  33      33  JJ      JJ  00      00  UU      UU  RR    RR    NN      NN  LL                ....
RR      RR  MM      MM    333333      JJJJJJ      000000    UUUUUUUUUU  RR      RR  NN      NN  LLLLLLLLLL        ....
RR      RR  MM      MM    333333      JJJJJJ      000000    UUUUUUUUUU  RR      RR  NN      NN  LLLLLLLLLL        ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
;     1          0001  0 MODULE RM3JOURNL (LANGUAGE (BLISS32) ,
;     2          0002  0                    IDENT = 'V04-000'
;     3          0003  0                   ) =
;     4          0004  1 BEGIN
;     5          0005  1 !
;     6          0006  1 !*****************************************************************************
;     7          0007  1 !*                                                                           *
;     8          0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
;     9          0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
;    10          0010  1 !*  ALL RIGHTS RESERVED.                                                     *
;    11          0011  1 !*                                                                           *
;    12          0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;    13          0013  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    14          0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    15          0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    16          0016  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    17          0017  1 !*  TRANSFERRED.                                                             *
;    18          0018  1 !*                                                                           *
;    19          0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;    20          0020  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;    21          0021  1 !*  CORPORATION.                                                             *
;    22          0022  1 !*                                                                           *
;    23          0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;    24          0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
;    25          0025  1 !*                                                                           *
;    26          0026  1 !*                                                                           *
;    27          0027  1 !*****************************************************************************
;    28          0028  1
;    29          0029  1 !++
;    30          0030  1 !
;    31          0031  1 ! FACILITY:     RMS32 INDEX SEQUENTIAL FILE ORGANIZATION
;    32          0032  1 !
;    33          0033  1 ! ABSTRACT:     This module contains routine specific for Recovery Unit
;    34          0034  1 !               Journaling and RU rollback recovery of RMS32 ISAM files.
;    35          0035  1 !
;    36          0036  1 !
;    37          0037  1 ! ENVIRONMENT:
;    38          0038  1 !
;    39          0039  1 !               VAX/VMS OPERATING SYSTEM
;    40          0040  1 !
;    41          0041  1 !--
;    42          0042  1
;    43          0043  1 !
;    44          0044  1 ! AUTHOR:       Todd M. Katz            CREATION DATE:          08-Jan-82
;    45          0045  1 !
;    46          0046  1 ! MODIFIED BY:
;    47          0047  1 !
;    48          0048  1 !       V03-011 DAS0002         David Solomon           25-Mar-1984
;    49          0049  1 !               Fix broken branches.
;    50          0050  1 !
;    51          0051  1 !       V03-010 DAS0001         David Solomon           01-Jul-1983
;    52          0052  1 !               Fill in correct value for RJR$B_ENTRY_TYPE.
;    53          0053  1 !
;    54          0054  1 !       V03-009 TSK0001         Tamar Krichevsky         7-Jun-1983
;    55          0055  1 !               Move module to RMS$RMS_JOURNAL psect.  Replace JNLDEF.R32
;    56          0056  1 !               with RMSINTDEF.L32.  Change addressing mode of RMS$RU_RECLAIM
;    57          0057  1 !               to long relative.
```

```
;  58          0058  1 ;
;  59          0059  1 ;       V03-008 KPL0001         Peter Lieberwirth       26-May-1983
;  60          0060  1 ;               New format of RJR.
;  61          0061  1 ;
;  62          0062  1 ;       V03-007 TMK0003         Todd M. Katz            03-Apr-1983
;  63          0063  1 ;               Whenever refering to the actual bucket contents being journalled
;  64          0064  1 ;               in RM$AI_AND_BI_3, refer to the bucket in the journalling buffer
;  65          0065  1 ;               and not to the bucket controlled by the arguement BDB. Note that
;  66          0066  1 ;               in the case of AI Journalling, these two buckets will be the
;  67          0067  1 ;               same, but this will not be so in the case of BI Journalling.
;  68          0068  1 ;
;  69          0069  1 ;       V03-006 TMK0003         Todd M. Katz            27-Mar-1983
;  70          0070  1 ;
;  71          0071  1 ;               1. Change the linkage of RM$RU_JOURNAL3 to RL$RABREG_67.
;  72          0072  1 ;               2. Change the linkage of RM$WRTJNL to RL$RABREG_4.
;  73          0073  1 ;               3. Change the routine RM$RU_JOURNAL3 to reflect the linkage
;  74          0074  1 ;                  changes.
;  75          0075  1 ;               4. Add the routine RM$AI_AND_BI_3 to direct the construction
;  76          0076  1 ;                  and journalling of entries to AI and BI Journals for ISAM
;  77          0077  1 ;                  files.
;  78          0078  1 ;               5. Modify RM$RU_RECOVER so that the RFA field within the RAB
;  79          0079  1 ;                  is not zeroed when the operation being recovered is a $FIND.
;  80          0080  1 ;
;  81          0081  1 ;       V03-005 MCN0002         Maria del C. Nasr       24-Mar-1983
;  82          0082  1 ;               More linkages reorganization.
;  83          0083  1 ;
;  84          0084  1 ;       V03-004 TMK0002         Todd M. Katz            17-Mar-1983
;  85          0085  1 ;               Change RJR$_DELET to RJR$_DELETE and RJR$_UPDAT to RJR$_UPDATE.
;  86          0086  1 ;               Also, fix up the External Register Linkages in RM$RU_JOURNL3.
;  87          0087  1 ;
;  88          0088  1 ;       V03-003 TMK0002         Todd M. Katz            16-Mar-1983
;  89          0089  1 ;
;  90          0090  1 ;               1. Change all RMSR$_ symbols to be RJR$_ symbols.
;  91          0091  1 ;               2. Change RJR$B_OP RJR$B_ORG to RJR$B_OPER and RJR$B_ENTRY_TYPE
;  92          0092  1 ;                  respectively.
;  93          0093  1 ;               3. Change the linkage to RM$RU_JOURNAL3 so that the BDB is an
;  94          0094  1 ;                  external register.
;  95          0095  1 ;               4. The interface to RM$WRTJNL has changed. Reflect this change
;  96          0096  1 ;                  within RM$RU_JOURNAL3.
;  97          0097  1 ;
;  98          0098  1 ;       V03-002 TMK0001         Todd M. Katz            11-Mar-1983
;  99          0099  1 ;               If the primary data bucket has not been exclusively locked,
; 100          0100  1 ;               then RM$RU_RECLAIM returns 0 indicating that the record/RRV
; 101          0101  1 ;               could not be reclaimed.
; 102          0102  1 ;
; 103          0103  1 ;       V03-001 MCN0001         Maria del C. Nasr       24-Feb-1983
; 104          0104  1 ;               Reorganize linkages
; 105          0105  1 ;
; 106          0106  1 ;*****
; 107          0107  1
; 108          0108  1 LIBRARY 'RMSLIB:RMSINTDEF';
; 109          0109  1
; 110          0110  1 LIBRARY 'SYS$LIBRARY:LIB';
; 111          0111  1
; 112          0112  1 REQUIRE 'RMSSRC:RMSIDXDEF';
; 113          0177  1
; 114          0178  1 ! Define default PSECTs for code.
```

```
;  115     0179  1 !
;  116     0180  1 PSECT
;  117     0181  1     CODE = RM$RMS_JOURNAL(PSECT_ATTR),
;  118     0182  1     PLIT = RM$RMS_JOURNAL(PSECT_ATTR);
;  119     0183  1
;  120     0184  1 ! Linkages.
;  121     0185  1 !
;  122     0186  1 LINKAGE
;  123     0187  1     L_JSB,
;  124     0188  1     L_PRESERVE1,
;  125     0189  1     L_QUERY_AND_LOCK,
;  126     0190  1     L_RABREG,
;  127     0191  1     L_RABREG_4,
;  128     0192  1     L_RABREG_4567,
;  129     0193  1     L_RABREG_457,
;  130     0194  1     L_RABREG_567,
;  131     0195  1     L_RABREG_67,
;  132     0196  1     L_RABREG_7,
;  133     0197  1     L_REC_OVHD;
;  134     0198  1
;  135     0199  1
;  136     0200  1 ! External Routines.
;  137     0201  1 !
;  138     0202  1 EXTERNAL ROUTINE
;  139     0203  1     RM$DELETE3B         : RL$RABREG ADDRESSING_MODE (LONG_RELATIVE),
;  140     0204  1     RM$DELETE_UDR       : RL$RABREG_4567 ADDRESSING_MODE (LONG_RELATIVE),
;  141     0205  1     RM$KEY_DESC         : RL$RABREG_7 ADDRESSING_MODE (LONG_RELATIVE),
;  142     0206  1     RM$LOCR             : RL$QUERY_AND_LOCK ADDRESSING_MODE (LONG_RELATIVE),
;  143     0207  1     RM$MOVE             : RL$PRESERVE1 ADDRESSING_MODE (LONG_RELATIVE),
;  144     0208  1     RM$NOREAD_LONG      : RL$JSB ADDRESSING_MODE (LONG_RELATIVE),
;  145     0209  1     RM$QUERY_PROC       : RL$QUERY_AND_LOCK ADDRESSING_MODE (LONG_RELATIVE),
;  146     0210  1     RM$RECORD_ID        : RL$RABREG_67 ADDRESSING_MODE (LONG_RELATIVE),
;  147     0211  1     RM$RECORD_KEY       : RL$PRESERVE1 ADDRESSING_MODE (LONG_RELATIVE),
;  148     0212  1     RM$RECORD_VBN       : RL$PRESERVE1 ADDRESSING_MODE (LONG_RELATIVE),
;  149     0213  1     RM$REC_OVHD         : RL$REC_OVHD ADDRESSING_MODE (LONG_RELATIVE),
;  150     0214  1     RM$UPDATE3B         : RL$RABREG_67 ADDRESSING_MODE (LONG_RELATIVE),
;  151     0215  1     RM$WRTJNL           : RL$RABREG_4 ADDRESSING_MODE (LONG_RELATIVE);
;  152     0216  1
;  153     0217  1 ! Forward Routines.
;  154     0218  1 !
;  155     0219  1 FORWARD ROUTINE
;  156     0220  1     RM$RU_REFORMAT      : RL$RABREG_567 NOVALUE;
```

```
;  158          0221  1 %SBTTL 'RM$AI_AND_BI_3'
;  159          0222  1 GLOBAL ROUTINE RM$AI_AND_BI_3 (JOURNAL) : RL$RABREG_4 =
;  160          0223  1
;  161          0224  1 !++
;  162          0225  1 !
;  163          0226  1 ! FUNCTIONAL DESCRIPTION:
;  164          0227  1 !
;  165          0228  1 !       The purpose of this routine is to construct all AI and BI Journal
;  166          0229  1 !       entries for ISAM files, and to oversee their writing.
;  167          0230  1 !
;  168          0231  1 ! CALLING SEQUENCE:
;  169          0232  1 !
;  170          0233  1 !       RM$AI_AND_BI_3()
;  171          0234  1 !
;  172          0235  1 ! INPUT PARAMETERS:
;  173          0236  1 !
;  174          0237  1 !       JOURNAL                 - type of journalling being done (AI or BI)
;  175          0238  1 !
;  176          0239  1 ! IMPLICIT INPUT:
;  177          0240  1 !
;  178          0241  1 !       BDB                     - address of BDB for bucket to be Journalled
;  179          0242  1 !           BDB$L_ADDR          - address of buffer
;  180          0243  1 !           BDB$L_AI_BDB        - address of AI Journalling BDB
;  181          0244  1 !           BDB$L_BI_BDB        - address of BI Journalling BDB
;  182          0245  1 !           BDB$W_NUMB          - number of bytes of buffer in use
;  183          0246  1 !           BDB$L_VBN           - VBN of bucket
;  184          0247  1 !
;  185          0248  1 ! OUTPUT PARAMETER:
;  186          0249  1 !       NONE
;  187          0250  1 !
;  188          0251  1 ! IMPLICIT OUTPUT:
;  189          0252  1 !       NONE
;  190          0253  1 !
;  191          0254  1 ! ROUTINE VALUE:
;  192          0255  1 !
;  193          0256  1 !       Whatever value is returned from the call to RM$WRTJNL.
;  194          0257  1 !
;  195          0258  1 ! SIDE EFFECTS:
;  196          0259  1 !       NONE
;  197          0260  1 !
;  198          0261  1 !--
;  199          0262  1
;  200          0263  2     BEGIN
;  201          0264  2
;  202          0265  2     EXTERNAL REGISTER
;  203          0266  2         COMMON_RAB_STR,
;  204          0267  2         R_BDB_STR;
;  205          0268  2
;  206          0269  2     GLOBAL REGISTER
;  207          0270  2         RJR_ADDR = 5    : REF BBLOCK;
;  208          0271  2
;  209          0272  2     LOCAL
;  210          0273  2         JNL_BDB         : REF BBLOCK,
;  211          0274  2         RJR_BUCKET      : REF BBLOCK;
;  212          0275  2
;  213          0276  2     ! Retrieve the address of the appropriate journalling BDB, and then the
;  214          0277  2     ! appropriate journalling buffer from the journalling BDB.
```

```
;  215          0278  2      !
;  216          0279  2      IF .JOURNAL EQLU CJF$_AI
;  217          0280  2      THEN
;  218          0281  2          JNL_BDB = .BDB[BDB$L_AI_BDB]
;  219          0282  2      ELSE
;  220          0283  2          JNL_BDB = .BDB[BDB$L_BI_BDB];
;  221          0284  2
;  222          0285  2      RJR_ADDR    = .JNL_BDB[BDB$L_ADDR];
;  223          0286  2      RJR_BUCKET  = .RJR_ADDR + RJR$C_BKTLEN;
;  224          0287  2
;  225          0288  2      ! Construct the AI/BI Journal Entry for the current Journalled operation.
;  226          0289  2      ! If the bucket is a single block in size, or is an index bucket, then the
;  227          0290  2      ! entire bucket is journalled; otherwise, just the contents of the bucket
;  228          0291  2      ! up to the freespace pointer is journalled.
;  229          0292  2      !
;  230          0293  2      RJR_ADDR[RJR$B_ORG]          = RJR$C_IDX;
;  231          0294  2      RJR_ADDR[RJR$B_ENTRY_TYPE]   = RJR$C_BUCKET;
;  232          0295  2      RJR_ADDR[RJR$B_OPER]         = RJR$_BUCKET;
;  233          0296  2      RJR_ADDR[RJR$L_BKT_VBN]      = .BDB[BDB$L_VBN];
;  234          0297  2      RJR_ADDR[RJR$W_BKT_SIZE]     = .BDB[BDB$W_NUMB];
;  235          0298  2
;  236          0299  2      IF  .BDB[BDB$W_NUMB] EQLU 512
;  237          0300  2           OR
;  238          0301  2          .RJR_BUCKET[BKT$B_LEVEL] GTRU 0
;  239          0302  2      THEN
;  240          0303  2          RJR_ADDR[RJR$W_JBKT_SIZE] = .BDB[BDB$W_NUMB]
;  241          0304  2      ELSE
;  242          0305  2          RJR_ADDR[RJR$W_JBKT_SIZE] = .RJR_BUCKET[BKT$W_FREESPACE];
;  243          0306  2
;  244          0307  2      JNL_BDB[BDB$W_NUMB] = RJR$C_BKTLEN + .RJR_ADDR[RJR$W_JBKT_SIZE];
;  245          0308  2
;  246          0309  2      ! Write out the AI/BI Journal Entry, and return the success or status of
;  247          0310  2      ! the journal operation.
;  248          0311  2      !
;  249          0312  2      RETURN RMSWRTJNL (.JOURNAL, .JNL_BDB);
;  250          0313  2
;  251          0314  1      END;

                                                .TITLE  RM3JOURNL
                                                .IDENT  \V04-000\

                                                .EXTRN  RMSDELETE3B, RMSDELETE_UDR
                                                .EXTRN  RMSKEY_DESC, RMSLOCK
                                                .EXTRN  RMSMOVE, RMSNOREAD_LONG
                                                .EXTRN  RMSQUERY_PROC, RMSRECORD_ID
                                                .EXTRN  RMSRECORD_KEY, RMSRECORD_VBN
                                                .EXTRN  RMSREC_OVHD, RMSUPDATE3B
                                                .EXTRN  RMSWRTJNL

                                                .PSECT  RMS$RMS_JOURNAL,NOWRT,  GBL,  PIC,2

                          55  DD 00000 RMS$AI_AND_BI_3::
                                                PUSHL   R5                                      ; 0222
                03        08  AE  D1 00002     CMPL    JOURNAL, #3                             ; 0279
                              06  12 00006     BNEQ    1$                                      ;
                50        34  A4  D0 00008     MOVL    52(BDB), JNL_BDB                        ; 0281
```

```
                                   04  11 0000C        BRB     2$                                          ;
                  50          30   A4  D0 0000E 1$:    MOVL    48(BDB), JNL_BDB                            ; 0283
                  55          18   A0  D0 00012 2$:    MOVL    24(JNL_BDB), RJR_ADDR                       ; 0285
                  51          44   A5  9E 00016        MOVAB   68(R5), RJR_BUCKET                          ; 0286
             03   A5        0204   8F  B0 0001A        MOVW    #516, 3(RJR_ADDR)                           ; 0294
             05   A5          01   90 00020            MOVB    #1, 5(RJR_ADDR)                             ; 0295
             3C   A5          1C   A4  D0 00024        MOVL    28(BDB), 60(RJR_ADDR)                       ; 0296
             40   A5          14   A4  B0 00029        MOVW    20(BDB), 64(RJR_ADDR)                       ; 0297
           0200   8F          14   A4  B1 0002E        CMPW    20(BDB), #512                               ; 0299
                                   05  13 00034        BEQL    3$                                          ;
                              0C   A1  95 00036        TSTB    12(RJR_BUCKET)                              ; 0301
                                   07  13 00039        BEQL    4$                                          ;
             42   A5          14   A4  B0 0003B 3$:    MOVW    20(BDB), 66(RJR_ADDR)                       ; 0303
                                   05  11 00040        BRB     5$                                          ;
             42   A5          04   A1  B0 00042 4$:    MOVW    4(RJR_BUCKET), 66(RJR_ADDR)                 ; 0305
  14   A0    42   A5        0044   8F  A1 00047 5$:    ADDW3   #68, 66(RJR_ADDR), 20(JNL_BDB)              ; 0307
                                   50  DD 0004F        PUSHL   JNL_BDB                                     ; 0312
                              0C   AE  DD 00051        PUSHL   JOURNAL                                     ;
                       00000000G   EF  16 00054        JSB     RM$WRTJNL                                   ;
                  5E               08  C0 0005A        ADDL2   #8, SP                                      ;
                                   20  BA 0005D        POPR    #^M<R5>                                     ; 0314
                                       05 0005F        RSB                                                 ;

; Routine Size:  96 bytes,    Routine Base:  RM$RMS_JOURNAL + 0000
```

```
;  253          0315  1 %SBTTL 'RM$RU_JOURNAL3'
;  254          0316  1 GLOBAL ROUTINE RM$RU_JOURNAL3 (OPERATION, VBN, ID, SIZE) : RL$RABREG_67 =
;  255          0317  1
;  256          0318  1 !++
;  257          0319  1 !
;  258          0320  1 ! FUNCTIONAL DESCRIPTION:
;  259          0321  1 !
;  260          0322  1 !       The purpose of this routine is to construct all RU Journal entries
;  261          0323  1 !       for ISAM files, and to oversee their writing.
;  262          0324  1 !
;  263          0325  1 ! CALLING SEQUENCE:
;  264          0326  1 !
;  265          0327  1 !       RM$RU_JOURNAL3()
;  266          0328  1 !
;  267          0329  1 ! INPUT PARAMETERS:
;  268          0330  1 !
;  269          0331  1 !       OPERATION               - operation being RU Journalled
;  270          0332  1 !       VBN                     - VBN of RU Journalled record's RFA
;  271          0333  1 !       ID                      - ID of RU Journalled record's RFA
;  272          0334  1 !       SIZE                    - size of record image to be journalled
;  273          0335  1 !
;  274          0336  1 ! IMPLICIT INPUT:
;  275          0337  1 !
;  276          0338  1 !       IRAB                    - address of IRAB
;  277          0339  1 !           IRB$L_CURBDB        - address of BDB for primary data bucket
;  278          0340  1 !           IRB$L_JNLBDB        - address of BDB for journal entry buffer
;  279          0341  1 !
;  280          0342  1 !       REC_ADDR                - address of record image to be journalled
;  281          0343  1 !
;  282          0344  1 ! OUTPUT PARAMETER:
;  283          0345  1 !       NONE
;  284          0346  1 !
;  285          0347  1 ! IMPLICIT OUTPUT:
;  286          0348  1 !       NONE
;  287          0349  1 !
;  288          0350  1 ! ROUTINE VALUE:
;  289          0351  1 !
;  290          0352  1 !       whatever value is returned from the call to RM$WRTJNL.
;  291          0353  1 !
;  292          0354  1 ! SIDE EFFECTS:
;  293          0355  1 !       NONE
;  294          0356  1 !
;  295          0357  1 !--
;  296          0358  1
;  297          0359  2     BEGIN
;  298          0360  2
;  299          0361  2     EXTERNAL REGISTER
;  300          0362  2         COMMON_RAB_STR,
;  301          0363  2         R_REC_ADDR;
;  302          0364  2
;  303          0365  2     GLOBAL REGISTER
;  304          0366  2         RJR_ADDR = 5    : REF BBLOCK;
;  305          0367  2
;  306          0368  2     LOCAL
;  307          0369  2         JNL_BDB         : REF BBLOCK;
;  308          0370  2
;  309          0371  2     ! Retrieve the address of the RU Journal Entry buffer.
```

```
;  310     0372  2     !
;  311     0373  2     JNL_BDB = .IRAB[IRB$L_JNLBDB];
;  312     0374  2     RJR_ADDR = .JNL_BDB[BDB$L_ADDR];
;  313     0375  2
;  314     0376  2     ! Construct the RU Journal Entry for the current RU Journalled operation.
;  315     0377  2     !
;  316     0378  2     RJR_ADDR[RJR$B_ENTRY_TYPE]  = RJR$C_RECORD;
;  317     0379  2     RJR_ADDR[RJR$B_ORG]         = RJR$C_IDX;
;  318     0380  2     RJR_ADDR[RJR$B_OPER]        = .OPERATION;
;  319     0381  2     RJR_ADDR[RJR$L_RFA0]        = .VBN;
;  320     0382  2     RJR_ADDR[RJR$W_RFA4]        = .ID;
;  321     0383  2     RJR_ADDR[RJR$W_RSIZE]       = .SIZE;
;  322     0384  2
;  323     0385  2     BEGIN
;  324     0386  2
;  325     0387  2     GLOBAL REGISTER
;  326     0388  2         R_BDB,
;  327     0389  2         R_IDX_DFN;
;  328     0390  2
;  329     0391  2     JNL_BDB[BDB$W_NUMB] = RM$MOVE (.SIZE, .REC_ADDR, RJR_ADDR[RJR$T_RIMAGE])
;  330     0392  2                                 - .RJR_ADDR;
;  331     0393  2     END;
;  332     0394  2
;  333     0395  2     ! Write out the RU Journal Entry, and return the success or status of the
;  334     0396  2     ! journal operation.
;  335     0397  2     !
;  336     0398  2     BEGIN
;  337     0399  2
;  338     0400  2     GLOBAL REGISTER
;  339     0401  2         R_BDB;
;  340     0402  2
;  341     0403  2     BDB = .IRAB[IRB$L_CURBDB];
;  342     0404  2
;  343     0405  3     RETURN RM$WRTJNL (CJF$_RU, .JNL_BDB);
;  344     0406  2     END;
;  345     0407  2
;  346     0408  1     END;


                                00B0    8F   BB 00000 RM$RU_JOURNAL3::
                                                     PUSHR   #^M<R4,R5,R7>                          ; 0316
                          51      30    A9   D0 00004         MOVL    48(IRAB), JNL_BDB             ;  0373
                          55      18    A1   D0 00008         MOVL    24(JNL_BDB), RJR_ADDR         ;  0374
                    03    A5    0202    8F   B0 0000C         MOVW    #514, 3(RJR_ADDR)             ;  0378
                    05    A5      10    AE   90 00012         MOVB    OPERATION, 5(RJR_ADDR)        ;  0380
                    40    A5      14    AE   D0 00017         MOVL    VBN, 64(RJR_ADDR)             ;  0381
                    44    A5      18    AE   B0 0001C         MOVW    ID, 68(RJR_ADDR)              ;  0382
                    46    A5      1C    AE   B0 00021         MOVW    SIZE, 70(RJR_ADDR)            ;  0383
                                  48    A5   9F 00026         PUSHAB  72(RJR_ADDR)                  ;  0391
                                        56   DD 00029         PUSHL   REC_ADDR
                                        24   AE   DD 0002B     PUSHL   SIZE
                                00000000G    EF   16 0002E     JSB     RM$MOVE
                          5E                 08   C0 00034     ADDL2   #8, SP
                14    A1    50              55   A3 00037     SUBW3   RJR_ADDR, R0, 20(JNL_BDB)      ; 0392
```

```
                        54      20  A9  D0 0003C       MOVL    32(IRAB), BDB                   ; 0403
                        6E          51  D0 00040       MOVL    JNL_BDB, (SP)                   ; 0405
                                    01  DD 00043       PUSHL   #1                              ;
                            00000000G   EF  16 00045    JSB     RM$WRTJNL                       ;
                        5E              08  C0 0004B   ADDL2   #8, SP                         ;
                                00B0    8F  BA 0004E    POPR    #^M<R4,R5,R7>                   ; 0408
                                        05  00052       RSB                                     ;

; Routine Size:  83 bytes,    Routine Base:  RM$RMS_JOURNAL + 0060
```

```
;  348     0409  1 %SBTTL 'RMSRU_RECLAIM'
;  349     0410  1 GLOBAL ROUTINE RMSRU_RECLAIM : RL$RABREG_67 =
;  350     0411  1
;  351     0412  1 !++
;  352     0413  1 !
;  353     0414  1 ! FUNCTIONAL DESCRIPTION:
;  354     0415  1 !
;  355     0416  1 !       The purpose of this routine is to try and reclaim space from the current
;  356     0417  1 !       record which has been previously modified within a Recovery Unit. Such
;  357     0418  1 !       reclamation can only take place if the Recovery Unit in which the
;  358     0419  1 !       current record was modified has successfully terminated, the file
;  359     0420  1 !       has been opened for write access, and the primary data bucket containing
;  360     0421  1 !       the record has been exclusively locked.
;  361     0422  1 !
;  362     0423  1 !       If the current record was updated within a Recovery Unit that has since
;  363     0424  1 !       terminated, then at this time the record maybe re-formatted. This
;  364     0425  1 !       involves placing the record into the normal format from the special
;  365     0426  1 !       format it is put in to reserve space during a Recovery Unit, and
;  366     0427  1 !       reclamating any unused space.
;  367     0428  1 !
;  368     0429  1 !       If the current record was deleted within a Recovery Unit that has since
;  369     0430  1 !       terminated, then at this time the record is deleted for good according
;  370     0431  1 !       to the normal rules of primary data record or RRV deletion.
;  371     0432  1 !
;  372     0433  1 !       Note that if the record had both been deleted and updated within a
;  373     0434  1 !       Recovery Unit, then the deletion takes precedence over the updating.
;  374     0435  1 !
;  375     0436  1 !       This routine returns success whenever it has modified the current
;  376     0437  1 !       primary data record regardless of whether or not any space was actually
;  377     0438  1 !       reclaimed through doing do.
;  378     0439  1 !
;  379     0440  1 ! CALLING SEQUENCE:
;  380     0441  1 !
;  381     0442  1 !       RMSRU_RECLAIM()
;  382     0443  1 !
;  383     0444  1 ! INPUT PARAMETERS:
;  384     0445  1 !       NONE
;  385     0446  1 !
;  386     0447  1 ! IMPLICIT INPUT:
;  387     0448  1 !
;  388     0449  1 !       IFAB                    - address of IFAB
;  389     0450  1 !           IFB$V_RUP           - if set, Recovery Unit is in progress
;  390     0451  1 !           IFB$V_WRTACC        - if set, file is opened for write access
;  391     0452  1 !
;  392     0453  1 !       IRAB                    - address of IRAB
;  393     0454  1 !           IRB$L_CURBDB        - address of BDB for primary data bucket
;  394     0455  1 !
;  395     0456  1 !       REC_ADDR                - address of current primary data record
;  396     0457  1 !
;  397     0458  1 ! OUTPUT PARAMETER:
;  398     0459  1 !       NONE
;  399     0460  1 !
;  400     0461  1 ! IMPLICIT OUTPUT:
;  401     0462  1 !       NONE
;  402     0463  1 !
;  403     0464  1 ! ROUTINE VALUE:
;  404     0465  1 !
```

```
;  405     0466  1 !       0       - reclamation of the record was not possible.
;  406     0467  1 !       1       - reclamation of the record was possible.
;  407     0468  1 !       RLK     - reclamation of the record was not possible because it could
;  408     0469  1 !                 not be locked
;  409     0470  1 !
;  410     0471  1 ! SIDE EFFECTS:
;  411     0472  1 !
;  412     0473  1 !       If the current record had been updated within a Recovery Unit,
;  413     0474  1 !           then it might have been re-formatted.
;  414     0475  1 !       If the current record had been deleted within a Recovery Unit,
;  415     0476  1 !           then it might have been deleted for good and its space partially
;  416     0477  1 !           or totally reclaimed.
;  417     0478  1 !       If any reclamation took place, the BDB for the primary data bucket is
;  418     0479  1 !           marked dirty.
;  419     0480  1 !--
;  420     0481  1
;  421     0482  1     BEGIN
;  422     0483  2
;  423     0484  2     BUILTIN
;  424     0485  2         AP;
;  425     0486  2
;  426     0487  2     EXTERNAL REGISTER
;  427     0488  2         COMMON_RAB_STR,
;  428     0489  2         R_IDX_DFN_STR,
;  429     0490  2         R_REC_ADDR_STR;
;  430     0491  2
;  431     0492  2     GLOBAL REGISTER
;  432     0493  2         R_BDB;
;  433     0494  2
;  434     0495  2     LABEL
;  435     0496  2         RECLAIM;
;  436     0497  2
;  437     0498  2     LOCAL
;  438     0499  2         STATUS;
;  439     0500  2
;  440     0501  2     ! Determine the lock status of the record which has been modified
;  441     0502  2     ! with a Recovery Unit but do not wait for the lock to be released if
;  442     0503  2     ! another stream has the record locked.
;  443     0504  2     !
;  444     0505  2     AP  = 3;
;  445     0506  2     IRAB[IRB$V_NO_Q_WAIT] = 1;
;  446     0507  2     STATUS = RM$QUERY_PROC (RM$RECORD_VBN(), RM$RECORD_ID());
;  447     0508  2
;  448     0509  2     ! If and only if the Recovery Unit in which the current primary data record
;  449     0510  2     ! was modified has completed and the file was opened for write access can
;  450     0511  2     ! this record be subject to special processing. If the query lock indicates
;  451     0512  2     ! that the current record is not locked by any stream, or if the current
;  452     0513  2     ! stream already has the record locked but it is not in a Recovery Unit,
;  453     0514  2     ! then RMS may conclude that the Recovery Unit in which the current record
;  454     0515  2     ! was modified has concluded, and subject the record to special processing.
;  455     0516  2     !
;  456     0517  2     IF  .IFAB[IFB$V_WRTACC]
;  457     0518  2          AND
;  458     0519  2         (.STATUS<0,16> EQLU RMSSUC()
;  459     0520  3                 OR
;  460     0521  4                 (NOT .IFAB[IFB$V_RUP]
```

```
;  462     0523  4                            AND
;  463     0524  3                           .STATUS<0,16> EQLU RMSSUC(OK_ALK)))
;  464     0525  2          THEN
;  465     0526  2      RECLAIM:
;  466     0527  3              BEGIN
;  467     0528  3
;  468     0529  3              GLOBAL REGISTER
;  469     0530  3                  COMMON_IO_STR;
;  470     0531  3
;  471     0532  3              ! If the primary data bucket containing the record has not been
;  472     0533  3              ! exclusively locked, then no space reclamation can take place.
;  473     0534  3              !
;  474     0535  3              BDB = .IRAB[IRB$L_CURBDB];
;  475     0536  3
;  476     0537  3              IF NOT .BBLOCK[.BDB[BDB$L_BLB_PTR], BLB$V_LOCK]
;  477     0538  3              THEN
;  478     0539  4                  BEGIN
;  479     0540  4                  STATUS = 0;
;  480     0541  4                  LEAVE RECLAIM;
;  481     0542  3                  END;
;  482     0543  3
;  483     0544  3              ! Retrieve the address of the primary data bucket.
;  484     0545  3              !
;  485     0546  3              BKT_ADDR = .BDB[BDB$L_ADDR];
;  486     0547  3
;  487     0548  3              ! A 1 will be returned as the value of this routine indicating that
;  488     0549  3              ! reclamation was possible. This will be regardless of whether any
;  489     0550  3              ! space will actually be reclaimed. Also, mark the primary data bucket's
;  490     0551  3              ! BDB as dirty.
;  491     0552  3              !
;  492     0553  3              STATUS = 1;
;  493     0554  3              BDB[BDB$V_DRT] = 1;
;  494     0555  3
;  495     0556  3              ! If the current record had been deleted within a Recovery Unit then
;  496     0557  3              ! it maybe truely deleted at this time, and the space it occupies
;  497     0558  3              ! reclaimed according to the normal rules for the deletion of primary
;  498     0559  3              ! data or RRV records.
;  499     0560  3              !
;  500     0561  3              IF .REC_ADDR[IRC$V_RU_DELETE]
;  501     0562  3              THEN
;  502     0563  4                  BEGIN
;  503     0564  4
;  504     0565  4                  ! Clear the RU_DELETE and the RU_UPDATE bit within the current
;  505     0566  4                  ! record's control byte.
;  506     0567  4                  !
;  507     0568  4                  REC_ADDR[IRC$V_RU_DELETE] = 0;
;  508     0569  4                  REC_ADDR[IRC$V_RU_UPDATE] = 0;
;  509     0570  4
;  510     0571  4                  ! Delete the current record (RRV or primary data record).
;  511     0572  4                  !
;  512     0573  4                  IF NOT .REC_ADDR[IRC$V_RRV]
;  513     0574  4                  THEN
;  514     0575  4                      RM$DELETE_UDR()
;  515     0576  4                  ELSE
;  516     0577  5                      BEGIN
;  517     0578  5
;  518     0579  5                      LOCAL
```

```
;   519          0580  5                      LENGTH;
;   520          0581  5
;   521          0582  6                  LENGTH = (.BKT_ADDR + .BKT_ADDR[BKT$W_FREESPACE])
;   522          0583  5                                   - (.REC_ADDR + IRC$C_FIXOVHSZ3);
;   523          0584  5
;   524          0585  5                  IF .LENGTH GTR 0
;   525          0586  5                  THEN
;   526          0587  5                      RMSMOVE (.LENGTH, .REC_ADDR + IRC$C_FIXOVHSZ3, .REC_ADDR);
;   527          0588  5
;   528          0589  5                  BKT_ADDR[BKT$W_FREESPACE] = .BKT_ADDR[BKT$W_FREESPACE]
;   529          0590  5                                           - IRC$C_FIXOVHSZ3;
;   530          0591  4                  END;
;   531          0592  4              END
;   532          0593  4
;   533          0594  4          ! If the current record had been updated within a Recovery Unit
;   534          0595  4          ! then it maybe reformated at this time.
;   535          0596  4          !
;   536          0597  4          ELSE
;   537          0598  4              RM$RU_REFORMAT();
;   538          0599  4          END
;   539          0600  3
;   540          0601  3      ! If RMS is unable to lock the current primary data record, or if the
;   541          0602  3      ! stream itself has it locked and the current process is within a Recovery
;   542          0603  3      ! Unit then RMS concludes that the Recovery Unit in which the record was
;   543          0604  3      ! modified has not successfully concluded. In these cases, and also when
;   544          0605  3      ! the file was not opned for write access the routine will return a status
;   545          0606  3      ! indicating that no reclamation was possible. RLK will be returned if RMS
;   546          0607  3      ! could not lock the record; otherwise, a status of 0 is returned.
;   547          0608  3      !
;   548          0609  2      ELSE
;   549          0610  3          IF .STATUS<0,16> NEQU RMSERR(RLK)
;   550          0611  2          THEN
;   551          0612  2              STATUS = 0;
;   552          0613  2
;   553          0614  2      ! Return whether or not any reclamation of the current primary data record
;   554          0615  2      ! was possible.
;   555          0616  2      !
;   556          0617  2      RETURN .STATUS;
;   557          0618  1      END;


                         3C  BB 00000 RM$RU_RECLAIM::
                                      PUSHR   #^M<R2,R3,R4,R5>                ; 0410
                         03  D0 00002          MOVL    #3, AP                  ; 0506
                      07 A9  01  88 00005          BISB2   #1, 7(IRAB)             ; 0507
                00000000G    EF  16 00009          JSB     RM$RECORD_ID            ; 0508
                         52  50  D0 0000F          MOVL    R0, R2                  ;
                00000000G    EF  16 00012          JSB     RM$RECORD_VBN           ;
                         51  50  D0 00018          MOVL    R0, R1                  ;
                00000000G    EF  16 0001B          JSB     RM$QUERY_PROC           ;
                         52  50  D0 00021          MOVL    R0, STATUS              ;
                      67 06  AA  E9 00024          BLBC    6(IFAB), 5$             ; 0518
                         01  52  B1 00028          CMPW    STATUS, #1              ; 0520
                             0D  13 0002B          BEQL    1$                      ;
```

```
                5C    00A2  CA              02  E0 0002D          BBS     #2, 162(IFAB), 5$                       ; 0522
                      8039  8F              52  B1 00033          CMPW    STATUS, #32825                          ; 0524
                                            55  12 00038          BNEQ    5$                                      ;
                            54        20      A9  D0 0003A 1$:      MOVL    32(IRAB), BDB                       ; 0535
                            50        10      A4  D0 0003E          MOVL    16(BDB), R0                         ; 0537
                            50        0A      A0  E9 00042          BLBC    10(R0), 6$                          ;
                            55        18      A4  D0 00046          MOVL    24(BDB), BKT_ADDR                   ; 0546
                            52                01  D0 0004A          MOVL    #1, STATUS                          ; 0553
                      0A    A4                02  88 0004D          BISB2   #2, 10(BDB)                         ; 0554
                35          66                05  E1 00051          BBC     #5, (REC_ADDR), 4$                  ; 0561
                            66        60      8F  8A 00055          BICB2   #96, (REC_ADDR)                     ; 0569
                08          66                03  E0 00059          BBS     #3, (REC_ADDR), 2$                  ; 0573
                                00000000G       EF  16 0005D          JSB     RM$DELETE_UDR                       ; 0575
                                                33  11 00063          BRB     7$                                  ;
                            50        04      A5  3C 00065 2$:      MOVZWL  4(BKT_ADDR), R0                     ; 0582
                            50                55  C0 00069          ADDL2   BKT_ADDR, R0                        ;
                            50                56  C2 0006C          SUBL2   REC_ADDR, R0                        ; 0583
                            50                09  C2 0006F          SUBL2   #9, LENGTH                          ;
                                              10  15 00072          BLEQ    3$                                  ; 0585
                                              56  DD 00074          PUSHL   REC_ADDR                            ; 0587
                                        09      A6  9F 00076          PUSHAB  9(R6)                               ;
                                              50  DD 00079          PUSHL   LENGTH                              ;
                                00000000G       EF  16 00078          JSB     RM$MOVE                             ;
                            5E                  0C  C0 00081          ADDL2   #12, SP                             ;
                      04    A5                  09  A2 00084 3$:      SUBW2   #9, 4(BKT_ADDR)                     ; 0590
                                                0E  11 00088          BRB     7$                                  ; 0561
                                      0000V     30  0008A 4$:         BSBW    RM$RU_REFORMAT                      ; 0598
                                                09  11 0008D          BRB     7$                                  ; 0518
                      82AA  8F                  52  B1 0008F 5$:      CMPW    STATUS, #33450                      ; 0610
                                                02  13 00094          BEQL    7$                                  ;
                                                52  D4 00096 6$:      CLRL    STATUS                              ; 0612
                            50                  52  D0 00098 7$:      MOVL    STATUS, R0                          ; 0617
                                                3C  BA 0009B          POPR    #^M<R2,R3,R4,R5>                    ; 0618
                                                    05 0009D          RSB                                         ;

; Routine Size:  158 bytes,    Routine Base:  RM$RMS_JOURNAL + 00B3
```

```
;  559     0619  1 %SBTTL 'RMSRU_RECOVER'
;  560     0620  1 GLOBAL ROUTINE RMSRU_RECOVER (OPERATION) : RL$RABREG =
;  561     0621  1
;  562     0622  1 !++
;  563     0623  1 !
;  564     0624  1 ! FUNCTIONAL DESCRIPTION:
;  565     0625  1 !
;  566     0626  1 !       The purpose of this routine is to oversee the RU ROLLBACK Recovery
;  567     0627  1 !       operations. Whenever one of these operations are initiated on an ISAM
;  568     0628  1 !       file, it is intercepted by the appropriate routine in the module
;  569     0629  1 !       RM3FACE, and control is transfered here. This routine then performs a
;  570     0630  1 !       number of checks, sets up the internal environment common to all RU
;  571     0631  1 !       ROLLBACK operations, and then dispatches to the code which actually
;  572     0632  1 !       directs each of the individual RU ROLLBACK Recovery operations.
;  573     0633  1 !
;  574     0634  1 ! CALLING SEQUENCE:
;  575     0635  1 !
;  576     0636  1 !       RMSRU_RECOVER()
;  577     0637  1 !
;  578     0638  1 ! INPUT PARAMETERS:
;  579     0639  1 !
;  580     0640  1 !       OPERATION               - the operation to be RU ROLLBACK Recovered
;  581     0641  1 !
;  582     0642  1 ! IMPLICIT INPUT:
;  583     0643  1 !
;  584     0644  1 !       IDX_DFN                 - address of the primary key index descriptor
;  585     0645  1 !           IDX$B_DATBKTSZ      - size of a primary data bucket in blocks
;  586     0646  1 !           IDX$B_DATBKTYP      - primary data bucket type
;  587     0647  1 !           IDX$V_DUPKEYS       - if set, duplicate primary keys are allowed
;  588     0648  1 !           IDX$V_KEY_COMPR     - if set, primary key compression is enabled
;  589     0649  1 !           IDX$B_KEYSZ         - size of primary key
;  590     0650  1 !           IDX$W_MINRECSZ      - minimum size of record to contain primary key
;  591     0651  1 !           IDX$V_REC_COMPR     - if set, record compression is enabled
;  592     0652  1 !
;  593     0653  1 !       IFAB                    - address of IFAB
;  594     0654  1 !           IFB$W_KBUFSZ        - size of an internal keybuffer
;  595     0655  1 !           IFB$L_LRL           - longest record length
;  596     0656  1 !           IFB$W_MRS           - maximum record size
;  597     0657  1 !           IFB$B_RFMORG        - record format
;  598     0658  1 !
;  599     0659  1 !       IRAB                    - address of IRAB
;  600     0660  1 !           IRB$L_KEYBUF        - address of the contigious keybuffers
;  601     0661  1 !           IRB$B_MODE          - access mode of the user operaion
;  602     0662  1 !
;  603     0663  1 !       RAB                     - address of the RAB
;  604     0664  1 !           RAB$L_RBF           - address of the user record buffer
;  605     0665  1 !           RAB$L_RFA0          - RFA VBN of the record to be RU Recovered
;  606     0666  1 !           RAB$W_RFA4          - RFA ID of the record to be RU Recovered
;  607     0667  1 !           RAB$W_RSZ           - size of the user record
;  608     0668  1 !
;  609     0669  1 ! OUTPUT PARAMETER:
;  610     0670  1 !       NONE
;  611     0671  1 !
;  612     0672  1 ! IMPLICIT OUTPUT:
;  613     0673  1 !
;  614     0674  1 !       IRAB                    - address of the IRAB
;  615     0675  1 !           IRB$B_CUR_KREF      - 0
```

```
;  616          0676  1 !           IRB$L_RBF           - address of the user record buffer
;  617          0677  1 !           IRB$B_RP_KREF       - 0
;  618          0678  1 !           IRB$W_RSZ           - size of the user record
;  619          0679  1 !           IRB$W_POS_ID        - RFA ID of the record to be RU Recovered
;  620          0680  1 !           IRB$L_POS_VBN       - RFA VBN of the record to be RU Recovered
;  621          0681  1 !           IRB$W_UDR_ID        - RFA ID of the record to be RU Recovered
;  622          0682  1 !           IRB$L_UDR_VBN       - RFA VBN of the record to be RU Recovered
;  623          0683  1 !
;  624          0684  1 !       RAB                     - address of user RAB
;  625          0685  1 !           RAB$L_RFA0          - 0 (Unless the operation is a $FIND Recovery)
;  626          0686  1 !           RAB$W_RFA4          - 0 (Unless the operation is a $FIND Recovery)
;  627          0687  1 !
;  628          0688  1 ! ROUTINE VALUE:
;  629          0689  1 !
;  630          0690  1 !       CUR - there is no current record to be RU ROLLBACK Recovered.
;  631          0691  1 !       RBF - unable to read user's record buffer.
;  632          0692  1 !       RSZ - user record size is bad
;  633          0693  1 !       SUC - successful RU ROLLBACK Recovery operatio.
;  634          0694  1 !
;  635          0695  1 !       Various Routine values from the following routines:
;  636          0696  1 !
;  637          0697  1 !       RM$DELETE3B
;  638          0698  1 !       RM$LOCK
;  639          0699  1 !       RM$UPDATE3B
;  640          0700  1 !
;  641          0701  1 ! SIDE EFFECTS:
;  642          0702  1 !
;  643          0703  1 !       On success, the RU operation will have been successfully recovered.
;  644          0704  1 !       On failures, the RU operation might have been successfully recovered
;  645          0705  1 !           depending on where the failure occurred and what the failure was.
;  646          0706  1 !
;  647          0707  1 !       AP is trashed.
;  648          0708  1 !       The primary key of the record will be placed into keybuffers 1 and 2.
;  649          0709  1 !       Several parts of the NRP context will be initialized with information
;  650          0710  1 !           about the record that is to be recovered.
;  651          0711  1 !       The RAB's RFA field will be zeroed (Unless the operation is a
;  652          0712  1 !           $FIND Recovery).
;  653          0713  1 !
;  654          0714  1 !
;  655          0715  1 !--
;  656          0716  1
;  657          0717  2     BEGIN
;  658          0718  2
;  659          0719  2     BUILTIN
;  660          0720  2         AP;
;  661          0721  2
;  662          0722  2     EXTERNAL REGISTER
;  663          0723  2         COMMON_RAB_STR;
;  664          0724  2
;  665          0725  2     LABEL
;  666          0726  2         IN.TIALIZE;
;  667          0727  2
;  668          0728  2     ! Perform the initilizations and checks common to all RU ROLLBACK Recovery
;  669          0729  2     ! operations.
;  670          0730  2     !
;  671          0731  2 INITIALIZE:
;  672          0732  3     BEGIN
```

```
;  673          0733  1
;  674          0734  1      GLOBAL REGISTER
;  675          0735  1          R_IDX_DFN_STR;
;  676          0736  1
;  677          0737  1      ! Make sure there is a record to be recovered.
;  678          0738  1      !
;  679          0739  1      IF  .RAB[RAB$L_RFA0] EQLU 0
;  680          0740  1       OR
;  681          0741  1          .RAB[RAB$W_RFA4] EQLU 0
;  682          0742  1      THEN
;  683          0743  1          RETURN RMSERR(CUR);
;  684          0744  1
;  685          0745  1      ! If this is a $FIND RU ROLLBACK Recovery operation than all the required
;  686          0746  1      ! initializations and checks have been performed.
;  687          0747  1      !
;  688          0748  1      IF .OPERATION EQLU RJR$_FIND
;  689          0749  1      THEN
;  690          0750  1          LEAVE INITIALIZE;
;  691          0751  1
;  692          0752  1      ! Save the size of the record and the address of the record buffer.
;  693          0753  1      !
;  694          0754  1      IRAB[IRB$L_RBF] = .RAB[RAB$L_RBF];
;  695          0755  1      IRAB[IRB$W_RSZ] = .RAB[RAB$W_RSZ];
;  696          0756  1
;  697          0757  1      ! Make sure the size of the record isn't greater than the maximum record
;  698          0758  1      ! size allowed.
;  699          0759  1      !
;  700          0760  1      IF .IFAB[IFB$B_RFMORG] EQL FAB$C_FIX
;  701          0761  1      THEN
;  702          0762  1          BEGIN
;  703          0763  1
;  704          0764  1          IF .IRAB[IRB$W_RSZ] NEQU .IFAB[IFB$W_LRL]
;  705          0765  1          THEN
;  706          0766  1              RETURN RMSERR(RSZ);
;  707          0767  1          END
;  708          0768  1      ELSE
;  709          0769  1          IF .IFAB[IFB$W_MRS] NEQ 0
;  710          0770  1              AND
;  711          0771  1              .IRAB[IRB$W_RSZ] GTRU .IFAB[IFB$W_MRS]
;  712          0772  1          THEN
;  713          0773  1              RETURN RMSERR(RSZ);
;  714          0774  1
;  715          0775  1      ! Make sure the record will fit in a primary data bucket. This is done
;  716          0776  1      ! by taking the size of the bucket less bucket overhead, subtracting the
;  717          0777  1      ! maximum overhead which maybe associated with a record in this file
;  718          0778  1      ! including possible key and record compression overhead, and comparing this
;  719          0779  1      ! value with the size of the record.
;  720          0780  1      !
;  721          0781  1      BEGIN
;  722          0782  1
;  723          0783  1      LOCAL
;  724          0784  1          BUCKET_SIZE     : WORD;
;  725          0785  1
;  726          0786  1      ! Retrieve the index descriptor for the primary key of reference.
;  727          0787  1      !
;  728          0788  1      RETURN_ON_ERROR (RM$KEY_DESC(0));
;  729          0789  1
```

```
;  730     0790  4     BUCKET_SIZE = (.IDX_DFN[IDX$B_DATBKTSZ] * 512) - BKT$C_OVERHDSZ
;  731     0791  4                                                    - BKT$C_DATBKTOVH
;  732     0792  4                                                    - IRC$C_FIXOVHSZ3;
;  733     0793  4
;  734     0794  4     IF .IDX_DFN[IDX$V_DUPKEYS]
;  735     0795  4     THEN
;  736     0796  4         BUCKET_SIZE = .BUCKET_SIZE - BKT$C_DUPBKTOVH;
;  737     0797  4
;  738     0798  4     IF  .IFAB[IFB$B_RFMORG] NEQU FAB$C_FIX
;  739     0799  4          OR
;  740     0800  5         (.IFAB[IFB$B_RFMORG] EQLU FAB$C_FIX
;  741     0801  5                  AND
;  742     0802  5                .IDX_DFN[IDX$B_DATBKTYP] NEQU IDX$C_NCMPNCMP)
;  743     0803  4     THEN
;  744     0804  4         BUCKET_SIZE = .BUCKET_SIZE - IRC$C_DATSZFLD;
;  745     0805  4
;  746     0806  4     IF .IDX_DFN[IDX$V_KEY_COMPR]
;  747     0807  4     THEN
;  748     0808  4         BUCKET_SIZE = .BUCKET_SIZE - IRC$C_KEYCMPOVH;
;  749     0809  4
;  750     0810  4     IF .IDX_DFN[IDX$V_REC_COMPR]
;  751     0811  4     THEN
;  752     0812  4         BUCKET_SIZE = .BUCKET_SIZE - IRC$C_DATCMPOVH;
;  753     0813  4
;  754     0814  4     IF .IRAB[IRB$W_RSZ] GTRU .BUCKET_SIZE
;  755     0815  4     THEN
;  756     0816  4         RETURN RMSERR(RSZ);
;  757     0817  3     END;
;  758     0818  3
;  759     0819  3     ! Verify that the record is large enough to contain the whole primary key.
;  760     0820  3     !
;  761     0821  3     IF .IRAB[IRB$W_RSZ] LSSU .IDX_DFN[IDX$W_MINRECSZ]
;  762     0822  3     THEN
;  763     0823  3         RETURN RMSERR(RSZ);
;  764     0824  3
;  765     0825  3     ! Probe the record buffer.
;  766     0826  3     !
;  767     0827  3     IF RMS$NOREAD_LONG (.IRAB[IRB$W_RSZ], .IRAB[IRB$L_RBF], .IRAB[IRB$B_MODE])
;  768     0828  3     THEN
;  769     0829  3         RETURN RMSERR(RBF);
;  770     0830  3
;  771     0831  3     ! Extract the primary key of the record into keybuffers 1 and 2.
;  772     0832  3     !
;  773     0833  4     BEGIN
;  774     0834  4
;  775     0835  4     GLOBAL REGISTER
;  776     0836  4         R_BDB,
;  777     0837  4         R_REC_ADDR;
;  778     0838  4
;  779     0839  4     AP = 3;
;  780     0840  4     REC_ADDR = .IRAB[IRB$L_RBF];
;  781     0841  4     RMS$RECORD_KEY (KEYBUF_ADDR(1));
;  782     0842  4
;  783     0843  4     RMS$MOVE (.IDX_DFN[IDX$B_KEYSZ], KEYBUF_ADDR(1), KEYBUF_ADDR(2));
;  784     0844  3     END;
;  785     0845  3
;  786     0846  3     ! Initialize the fields in the NRP such that the record being recovered
```

```
;  787     0847  3      ! becomes the current primary data record.
;  788     0848  3      !
;  789     0849  3      IRAB[IRB$B_CUR_KREF] = 0;
;  790     0850  3      IRAB[IRB$B_RP_KREF]  = 0;
;  791     0851  3
;  792     0852  3      IRAB[IRB$L_UDR_VBN]  = .RAB[RAB$L_RFA0];
;  793     0853  3      IRAB[IRB$W_UDR_ID]   = .RAB[RAB$W_RFA4];
;  794     0854  3      IRAB[IRB$L_POS_VBN]  = .IRAB[IRB$L_UDR_VBN];
;  795     0855  3      IRAB[IRB$W_POS_ID]   = .IRAB[IRB$W_UDR_ID];
;  796     0856  3      END;
;  797     0857  3
;  798     0858  3      ! Dispatch to the RU ROLLBACK Recovery Code Which is Specific for each
;  799     0859  3      ! type of operation to be recovered.
;  800     0860  3      !
;  801     0861  3      BEGIN
;  802     0862  3
;  803     0863  3      LOCAL
;  804     0864  3          STATUS;
;  805     0865  3
;  806     0866  3      SELECTONEU .OPERATION OF
;  807     0867  3          SET
;  808     0868  3
;  809     0869  3          ! The RU ROLLBACK operation requested is a $FIND. This just consists of
;  810     0870  3          ! locking the record with the indicated RFA.
;  811     0871  3          !
;  812     0872  3          [RJR$_FIND]:    STATUS = RM$LOCK (.RAB[RAB$L_RFA0], .RAB[RAB$W_RFA4]);
;  813     0873  3
;  814     0874  3          ! The RU ROLLBACK operation requested is a $DELETE. This Recovery
;  815     0875  3          ! operation consists of un-deleting each part of the current record that
;  816     0876  3          ! had been deleted within the Recovery Unit being rolled back.
;  817     0877  3          !
;  818     0878  4          [RJR$_DELETE]:  BEGIN
;  819     0879  4                          IRAB[IRB$V_RU_UNDEL] = 1;
;  820     0880  4                          STATUS = RM$DELETE3B();
;  821     0881  4                          IRAB[IRB$V_RU_UNDEL] = 0;
;  822     0882  4                          END;
;  823     0883  3
;  824     0884  3          ! The RU ROLLBACK operation requested is a $PUT. This Recovery operation
;  825     0885  3          ! consists of deleting each and every part of the current record that
;  826     0886  3          ! was inserted as a new record within the Recovery Unit being rolled
;  827     0887  3          ! back.
;  828     0888  3          !
;  829     0889  3          [RJR$_PUT]:     STATUS = RM$DELETE3B();
;  830     0890  3
;  831     0891  3          ! The RU ROLLBACK operation requested is a $UPDATE. This Recovery
;  832     0892  3          ! operation consists of replacing a newer version of a record with an
;  833     0893  3          ! older version of the same record which had been replaced within the
;  834     0894  3          ! Recovery Unit being rolled back.
;  835     0895  3          !
;  836     0896  4          [RJR$_UPDATE]:  BEGIN
;  837     0897  4
;  838     0898  4                          GLOBAL REGISTER
;  839     0899  4                              R_IDX_DFN,
;  840     0900  4                              R_REC_ADDR;
;  841     0901  4
;  842     0902  4                          IRAB[IRB$V_UPDATE] = 1;
;  843     0903  4                          STATUS = RM$UPDATE3B();
```

```
;  844     0904  4                        IRAB[IRB$V_UPDATE] = 1;
;  845     0905  4                        END;
;  846     0906                      TES;
;  847     0907
;  848     0908          ! Zero in the user's RAB the RFA of the record which has been RU ROLLBACK
;  849     0909          ! Recovered unless the operation being recovered is a $FIND.
;  850     0910          !
;  851     0911          IF .OPERATION NEQU RJR$_FIND
;  852     0912          THEN
;  853     0913  2           BEGIN
;  854     0914  2           RAB[RAB$W_RFA4] = 0;
;  855     0915  2           RAB[RAB$L_RFA0] = 0;
;  856     0916  2           END;
;  857     0917
;  858     0918          ! Return the status of the RU ROLLBACK Recovery operation.
;  859     0919          !
;  860     0920          RETURN .STATUS;
;  861     0921          END;
;  862     0922  1       END;


                                00FC   8F   BB 00000 RM$RU_RECOVER::
                                                     PUSHR   #^M<R2,R3,R4,R5,R6,R7>          ; 0620
                                  10   A8   D5 00004          TSTL    16(RAB)                 ; 0739
                                       05   13 00007          BEQL    1$
                                  14   A8   B5 00009          TSTW    20(RAB)                 ; 0741
                                       08   12 0000C          BNEQ    3$
                             50    84B4   8F   3C 0000E 1$:    MOVZWL  #33972, R0              ; 0743
                                        0143   31 00013 2$:    BRW     21$
                             55      1C   AE   D0 00016 3$:    MOVL    OPERATION, R5           ; 0748
                             0B           55   D1 0001A          CMPL    R5, #11
                                          03   12 0001D          BNEQ    4$
                                        00E0   31 0001F          BRW     16$
                          58  A9        28   A8   D0 00022 4$:  MOVL    40(RAB), 88(IRAB)       ; 0754
                          56  A9        22   A8   B0 00027      MOVW    34(RAB), 86(IRAB)       ; 0755
                              01        50   AA   91 0002C      CMPB    80(IFAB), #1            ; 0760
                                          09   12 00030        BNEQ    5$
                          52  AA        56   A9   B1 00032      CMPW    86(IRAB), 82(IFAB)      ; 0764
                                          0E   13 00037        BEQL    6$
                                          59   11 00039        BRB     12$                     ; 0766
                                    60    AA   B5 0003B 5$:    TSTW    96(IFAB)                ; 0769
                                          07   13 0003E        BEQL    6$
                          60  AA        56   A9   B1 00040      CMPW    86(IRAB), 96(IFAB)      ; 0771
                                          4D   1A 00045        BGTRU   12$
                                          7E   D4 00047 6$:    CLRL    -(SP)                   ; 0788
                                00000000G      EF   16 00049    JSB     RM$KEY_DESC
                                    5E         04   C0 0004F    ADDL2   #4, SP
                                    BE         50   E9 00052    BLBC    STATUS, 2$
                                    51   17    A7   9A 00055    MOVZBL  23(IDX_DFN), R1         ; 0790
                             51     51         09   78 00059    ASHL    #9, R1, R1
                             50     51         19   A3 0005D    SUBW3   #25, R1, BUCKET_SIZE    ; 0792
                                    03   1C    A7   E9 00061    BLBC    28(IDX_DFN), 7$         ; 0794
                                    50         04   A2 00065    SUBW2   #4, BUCKET_SIZE         ; 0796
                                    01    50   AA   91 00068 7$:  CMPB    80(IFAB), #1            ; 0798
```

```
                          06   12 0006C          BNEQ    8$                          ;
        06          29    A7   91 0006E          CMPB    41(IDX_DFN), #6             ; 0802
                          03   13 00072          BEQL    9$                          ;
        50                02   A2 00074 8$:      SUBW2   #2, BUCKET_SIZE             ; 0804
 03  1C A7                06   E1 00077 9$:      BBC     #6, 28(IDX_DFN), 10$        ; 0806
        50                02   A2 0007C          SUBW2   #2, BUCKET_SIZE             ; 0808
                    1C    A7   95 0007F 10$:     TSTB    28(IDX_DFN)                 ; 0810
                          03   18 00082          BGEQ    11$                         ;
        50                03   A2 00084          SUBW2   #3, BUCKET_SIZE             ; 0812
        50          56    A9   B1 00087 11$:     CMPW    86(IRAB), BUCKET_SIZE       ; 0814
                          07   1A 0008B          BGTRU   12$                         ;
     22 A7          56    A9   B1 0008D          CMPW    86(IRAB), 34(IDX_DFN)       ; 0821
                          07   1E 00092          BGEQU   13$                         ;
        50        86A4    8F   3C 00094 12$:     MOVZWL  #34468, R0                  ; 0823
                          1C   11 00099          BRB     14$                         ;
        7E          0A    A9   9A 0009B 13$:     MOVZBL  10(IRAB), -(SP)             ; 0827
                    58    A9   DD 0009F          PUSHL   88(IRAB)                    ;
        7E          56    A9   3C 000A2          MOVZWL  86(IRAB), -(SP)             ;
             00000000G    EF   16 000A6          JSB     RM$NOREAD_LONG              ;
        5E                0C   C0 000AC          ADDL2   #12, SP                     ;
        08                50   E9 000AF          BLBC    R0, 15$                     ;
        50        8654    8F   3C 000B2          MOVZWL  #34388, R0                  ; 0829
                        009F   31 000B7 14$:     BRW     21$                         ;
        5C                03   D0 000BA 15$:     MOVL    #3, AP                      ; 0839
        56          58    A9   D0 000BD          MOVL    88(IRAB), REC_ADDR          ; 0840
                    60    A9   DD 000C1          PUSHL   96(IRAB)                    ; 0841
             00000000G    EF   16 000C4          JSB     RM$RECORD_KEY               ;
        50        00B4    CA   3C 000CA          MOVZWL  180(IFAB), R0               ; 0843
        6E       60 B940  9E      000CF          MOVAB   @96(IRAB)[R0], (SP)         ;
                    60    A9   DD 000D4          PUSHL   96(IRAB)                    ;
        7E          20    A7   9A 000D7          MOVZBL  32(IDX_DFN), -(SP)          ;
             00000000G    EF   16 000DB          JSB     RM$MOVE                     ;
        5E                0C   C0 000E1          ADDL2   #12, SP                     ;
                  00C2    C9   B4 000E4          CLRW    194(IRAB)                   ; 0850
   00B0 C9          10    A8   D0 000E8          MOVL    16(RAB), 176(IRAB)          ; 0852
   00BC C9          14    A8   B0 000EE          MOVW    20(RAB), 188(IRAB)          ; 0853
   00AC C9        00B0    C9   D0 000F4          MOVL    176(IRAB), 172(IRAB)        ; 0854
   00BA C9        00BC    C9   B0 000FB          MOVW    188(IRAB), 186(IRAB)        ; 0855
        0B                55   D1 00102 16$:     CMPL    R5, #11                     ; 0872
                          10   12 00105          BNEQ    17$                         ;
        52          14    A8   3C 00107          MOVZWL  20(RAB), R2                 ;
        51          10    A8   D0 0010B          MOVL    16(RAB), R1                 ;
             00000000G    EF   16 0010F          JSB     RM$LOCK                     ;
                          37   11 00115          BRB     20$                         ;
        05                55   D1 00117 17$:     CMPL    R5, #5                      ; 0878
                          12   12 0011A          BNEQ    18$                         ;
     07 A9          40    8F   88 0011C          BISB2   #64, 7(IRAB)                ; 0879
             00000000G    EF   16 00121          JSB     RM$DELETE3B                 ; 0880
     07 A9          40    8F   8A 00127          BICB2   #64, 7(IRAB)                ; 0881
                          20   11 0012C          BRB     20$                         ; 0866
        13                55   D1 0012E 18$:     CMPL    R5, #19                     ; 0889
                          08   12 00131          BNEQ    19$                         ;
             00000000G    EF   16 00133          JSB     RM$DELETE3B                 ;
                          13   11 00139          BRB     20$                         ;
        1C                55   D1 0013B 19$:     CMPL    R5, #28                     ; 0896
                          0E   12 0013E          BNEQ    20$                         ;
     06 A9                08   88 00140          BISB2   #8, 6(IRAB)                 ; 0902
```

```
                          00000000G  EF  16 00144          JSB     RM$UPDATE3B                             ; 0903
                      06  A9         08  88 0014A          BISB2   #8, 6(IRAB)                           ; 0904
                          DB         55  D1 0014E 20$:      CMPL    R5, #11                                 ; 0911
                                     06  13 00151          BEQL    21$                                     ;
                                14   A8  B4 00153          CLRW    20(RAB)                                 ; 0914
                                10   A8  D4 00156          CLRL    16(RAB)                                 ; 0915
                               00FC  8F  BA 00159 21$:      POPR    #^M<R2,R3,R4,R5,R6,R7>                  ; 0922
                                     05  00159D           RSB                                             ;

; Routine Size:  350 bytes,    Routine Base:  RM$RMS_JOURNAL + 0151
```

```
;  864          0923  1 %SBTTL 'RM$RU_REFORMAT'
;  865          0924  1 GLOBAL ROUTINE RM$RU_REFORMAT : RL$RABREG_567 NOVALUE =
;  866          0925  1
;  867          0926  1 !++
;  868          0927  1 !
;  869          0928  1 ! FUNCTIONAL DESCRIPTION:
;  870          0929  1 !
;  871          0930  1 !       This routine's responsibility is to reformat primary data records which
;  872          0931  1 !       have decreased in size during an $UPDATE within a recovery unit and
;  873          0932  1 !       consequently were placed in a special format to reserve the space that
;  874          0933  1 !       would otherwise have been freed. Such records have the control bit
;  875          0934  1 !       IRC$V_RU_UPDATE set.
;  876          0935  1 !
;  877          0936  1 !       These records are in a special format in that two record sizes are
;  878          0937  1 !       associated with them. The number of bytes the primary data record
;  879          0938  1 !       reserves in the bucket (not including the record overhead) is stored
;  880          0939  1 !       in the record size field in the record overhead. The true size of the
;  881          0940  1 !       record is stored in the last two bytes of the primary data record.
;  882          0941  1 !
;  883          0942  1 !       This routine reformats the primary data record by:
;  884          0943  1 !
;  885          0944  1 !       1. Clearing the IRC$V_RU_UPDATE control bit.
;  886          0945  1 !       2. Moving the true record size into the record size field of the record
;  887          0946  1 !          overhead.
;  888          0947  1 !       3. Eliminating the space reserved by the record by shifting over the
;  889          0948  1 !          primary data records that follow it in the primary data bucket, so
;  890          0949  1 !          that this reserved space is freed.
;  891          0950  1 !       4. Adjusting the bucket's freespace offset pointer to reflect the bytes
;  892          0951  1 !          which have been freed through the reformating of the record.
;  893          0952  1 !
;  894          0953  1 ! CALLING SEQUENCE:
;  895          0954  1 !
;  896          0955  1 !       RM$RU_REFORMAT()
;  897          0956  1 !
;  898          0957  1 ! INPUT PARAMETERS:
;  899          0958  1 !       NONE
;  900          0959  1 !
;  901          0960  1 ! IMPLICIT INPUT:
;  902          0961  1 !
;  903          0962  1 !       BKT_ADDR                - address of the primary data bucket
;  904          0963  1 !           BKT$W_FREESPACE     - bucket's freespace offset pointer
;  905          0964  1 !
;  906          0965  1 !       REC_ADDR                - address of the record to be reformated
;  907          0966  1 !
;  908          0967  1 ! OUTPUT PARAMETER:
;  909          0968  1 !       NONE
;  910          0969  1 !
;  911          0970  1 ! IMPLICIT OUTPUT:
;  912          0971  1 !
;  913          0972  1 !       BKT_ADDR                - address of the primary data bucket
;  914          0973  1 !           BKT$W_FREESPACE     - bucket's freespace offset pointer
;  915          0974  1 !
;  916          0975  1 ! ROUTINE VALUE:
;  917          0976  1 !       NONE
;  918          0977  1 !
;  919          0978  1 ! SIDE EFFECTS:
;  920          0979  1 !
```

```
;  921     0980  1 !       The record is reformatted, and the bucket's freespace offset pointer
;  922     0981  1 !       is updated to reflect the bytes which have been freed.
;  923     0982  1 !
;  924     0983  1 !--
;  925     0984  1
;  926     0985  2     BEGIN
;  927     0986  2
;  928     0987  2     EXTERNAL REGISTER
;  929     0988  2         R_BKT_ADDR_STR,
;  930     0989  2         COMMON_RAB_STR,
;  931     0990  2         R_IDX_DFN,
;  932     0991  2         R_REC_ADDR_STR;
;  933     0992  2
;  934     0993  2     LOCAL
;  935     0994  2         FAKE_SIZE,
;  936     0995  2         LENGTH,
;  937     0996  2         SAVE_REC_ADDR,
;  938     0997  2         TRUE_SIZE;
;  939     0998  2
;  940     0999  2     ! Clear the special record format bit in the record's control byte.
;  941     1000  2     !
;  942     1001  2     REC_ADDR[IRC$V_RU_UPDATE] = 0;
;  943     1002  2
;  944     1003  2     ! Place the true size of the record in the record size field of the record
;  945     1004  2     ! overhead. This size maybe found in the last two bytes of the record proper
;  946     1005  2     ! as it currently exists in the primary data bucket.
;  947     1006  2     !
;  948     1007  3     BEGIN
;  949     1008  3
;  950     1009  3     LOCAL
;  951     1010  3         REC_SIZE;
;  952     1011  3
;  953     1012  3     SAVE_REC_ADDR       = .REC_ADDR;
;  954     1013  3     REC_ADDR            = .REC_ADDR + RM$REC_OVHD(0; REC_SIZE);
;  955     1014  3     FAKE_SIZE           = .REC_SIZE;
;  956     1015  2     END;
;  957     1016  2
;  958     1017  2     TRUE_SIZE = .(.REC_ADDR + .FAKE_SIZE - IRC$C_DATSZFLD)<0,16>;
;  959     1018  2
;  960     1019  2     (.REC_ADDR - IRC$C_DATSZFLD)<0,16> = .TRUE_SIZE;
;  961     1020  2
;  962     1021  2     ! If there are any records following the current record, shift them down
;  963     1022  2     ! in the primary data bucket so that the space, formerly reserved by this
;  964     1023  2     ! special record, is now utilized, and the corresponding amount of space
;  965     1024  2     ! is made available.
;  966     1025  2     !
;  967     1026  2     LENGTH = .BKT_ADDR[BKT$W_FREESPACE] - (.REC_ADDR + .FAKE_SIZE - .BKT_ADDR);
;  968     1027  2
;  969     1028  2     IF .LENGTH GTRU 0
;  970     1029  2     THEN
;  971     1030  3         BEGIN
;  972     1031  3
;  973     1032  3         GLOBAL REGISTER
;  974     1033  3             R_BDB;
;  975     1034  3
;  976     1035  3         RM$MOVE (.LENGTH, (.REC_ADDR + .FAKE_SIZE), (.REC_ADDR + .TRUE_SIZE));
;  977     1036  2         END;
```

```
;  978          1037  2
;  979          1038  2      ! Adjust the bucket's freespace offset pointer to reflect the amount of
;  980          1039  2      ! space which has become available through reformatting of the current
;  981          1040  2      ! record.
;  982          1041  2      !
;  983          1042  2      BKT_ADDR[BKT$W_FREESPACE] = .BKT_ADDR[BKT$W_FREESPACE]
;  984          1043  2                                - (.FAKE_SIZE - .TRUE_SIZE);
;  985          1044  2      REC_ADDR = .SAVE_REC_ADDR;
;  986          1045  1      END;


                                    1C  BB 00000 RM$RU_REFORMAT::
                                                 PUSHR   #^M<R2,R3,R4>                          ; 0924
                              5E        04  C2 00002         SUBL2   #4, SP
                              66  40    8F  8A 00005         BICB2   #64, (REC_ADDR)                ; 1001
                              53        56  D0 00009         MOVL    REC_ADDR, SAVE_REC_ADDR        ; 1012
                                        51  D4 0000C         CLRL    R1                             ; 1013
                           00000000G    EF  16 0000E         JSB     RM$REC_OVHD
                              56        50  C0 00014         ADDL2   R0, REC_ADDR
                              52        51  D0 00017         MOVL    REC_SIZE, FAKE_SIZE            ; 1014
                        54    56        52  C1 0001A         ADDL3   FAKE_SIZE, REC_ADDR, R4        ; 1017
                              51  FE    A4  3C 0001E         MOVZWL  -2(R4), TRUE_SIZE
                        FE    A6        51  B0 00022         MOVW    TRUE_SIZE, -2(REC_ADDR)        ; 1019
                        50    55        54  C3 00026         SUBL3   R4, BKT_ADDR, R0               ; 1026
                              6E  04    A5  3C 0002A         MOVZWL  4(BKT_ADDR), (SP)
                              50        6E  C0 0002E         ADDL2   (SP), LENGTH
                                        0E  13 00031         BEQL    1$                             ; 1028
                                      6146  9F 00033         PUSHAB  (TRUE_SIZE)[REC_ADDR]          ; 1035
                                        11  BB 00036         PUSHR   #^M<R0,R4>
                           00000000G    EF  16 00038         JSB     RM$MOVE
                              5E        0C  C0 0003E         ADDL2   #12, SP
                              51        52  C2 00041 1$:     SUBL2   FAKE_SIZE, R1                  ; 1043
                        04    A5        51  A0 00044         ADDW2   R1, 4(BKT_ADDR)
                              56        53  D0 00048         MOVL    SAVE_REC_ADDR, REC_ADDR        ; 1044
                              5E        04  C0 0004B         ADDL2   #4, SP                         ; 1045
                                        1C  BA 0004E         POPR    #^M<R2,R3,R4>
                                            05 00050         RSB

; Routine Size:  81 bytes,    Routine Base:  RM$RMS_JOURNAL + 02AF

;  987          1046  1
;  988          1047  1 END
;  989          1048  0 ELUDOM


;                                   PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
```

```
;  RM$RMS_JOURNAL                    768  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  GBL,  REL,  CON,  PIC,ALIGN(2)



;                              Library Statistics
;
;                                        --------- Symbols ---------      Pages       Processing
;       File                             Total     Loaded    Percent      Mapped      Time
;
;  _$255$DUA28:[RMS.OBJ]RMSINTDEF.L32;1    1484         74          4          83        00:00.2
;  _$255$DUA28:[SYSLIB]LIB.L32;1          18619         43          0        1000        00:04.6




;                                    COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:RM3JOURNL/OBJ=OBJ$:RM3JOURNL MSRC$:RM3JOURNL/UPDATE=(ENH$:RM3JOURNL)

; Size:         768 code + 0 data bytes
; Run Time:       00:25.3
; Elapsed Time:   00:48.7
; Lines/CPU Min:   2485
; Lexemes/CPU-Min: 14022
; Memory Used:  183 pages
; Compilation Complete
```

RM3MAKIDX
LIS

RM3FNDRRV
LIS

RM3FNDRFA
LIS

RM3GET
LIS

RM3ILUDR
LIS

RM3JOURNL
LIS

RM3KEYDSC
LIS

RM3MISC
LIS

RM3MISPUT
LIS